Latent Feature Estimation for Burak Model with EM

Alex Anderson[1] and Bruno Olshausen[1]

[1]University of California, Berkeley

July 18, 2015

Abstract

Humans can resolve the fine details of visual stimuli although the image projected on the retina is constantly drifting relative to the photoreceptor array. Further, we propose a decoding strategy for interpreting the spikes emitted by the retina, which takes into account the ambiguity caused by retinal noise and the unknown trajectory of the projected image on the retina. **?**. We use an EM algorithm to decode the retinal spikes. Unlike previous models, we assume that eye movements are continuous (as opposed to discrete) and pixel intensities are continuously valued (instead of binary). Furthermore, our algorithm allows for the incorporation of non-trivial image priors.

Contents

1 Introduction **4**

2 Model Specification **6**

3 Implementation Details **9**
3.1 Determining Image - Receptive Field Overlap 9
3.2 Kernel Trick to Reduce Memory 9

4 EM Algorithm **10**
4.1 EM Details 11
4.2 Particle Filtering 12
4.3 Creating images showing EM estimate improving over time 14
4.4 Optimizing the Objective Function with respect to the image 15
4.5 Optimizing the Objective Function 15
4.6 Choosing the Lipschitz Constant 16

5 Discrete Space EM Algorithm in Real-time **18**

6 Online EM Using Gaussian Estimation **21**
6.1 Justification of the Gaussian Approximation 23

7 Online EM with Sparse Prior and Gaussian Estimation **24**

8 Image Priors: Moving Beyond the Independent Pixel Assumption **25**
8.1 MNIST Prior 25
8.2 Natural Image Prior and Perceptually Relevant Cues 26

9 Motion Prior: Including Momentum and Acceleration **28**
9.1 Why include a motion prior? 28
9.2 A simple model of motion that can be integrated into the current framework 28
9.3 Incorporating the motion prior into the existing model 30
9.4 Fitting the model to the data 30
9.5 Validating the Model 30
9.6 Fitting Different Motion Models to Actual Trajectories 30

10 Drift in the presence of an inhomogeneous retina **32**

11 Common Questions and Subtle Concepts **33**
11.1 Perceptually Relevant Cues versus Pixel Reconstruction 33
11.2 Parsimonious versus Biophysically Realistic Modeling when making predictions 33
11.3 Optimality of a coding strategy versus experimental test of the use of such a strategy 33
11.4 Biological Implementation versus Implementation suggested by the model . . 33

12 Comparisons to Alternative Models **35**
12.1 Burak Model 35
12.2 Spike Averaging 35
12.3 Correlations 36
12.4 Comparing Diffusion Coefficients Across Different Models 37

13 Comparisons to Experiment **37**

14 Model Choice Discussion **38**
14.1 Image to Spike Encoding Model 38
14.2 Eye movement modeling 40
14.3 Decoding Method 40

15 Lessons Learned from Bayesian Inference **42**
15.1 Dynamic Routing of Spikes to form Image Estimate 42
15.2 Propagating the position estimates 42
15.3 Propagation of uncertainty is crucial in the presence of ambiguous signals . . 43
15.4 Details of the Image Representation are important 43

16 Additional questions to investigate **44**

17 MAP Estimation Results **45**
17.1 Expectation values of the MAP Method 45
17.2 Rate Coding as dt goes to zero 46
17.3 Rate coding with one neuron and one pixel 46
17.4 Normalization 47

18 Corrections **48**

1 Introduction

Our brains take in sensory data from our neurons and infer perceptually relevant features of the world. In the case of high acuity vision in humans, there are two important sources of noise: limited coding capacity of rate coded responses of retinal ganglion cells and random eye movements that cause the retinal image to jitter.

Humans with normal 20-20 vision are able to resolve visual features that differ by just a few photoreceptors (Eg. an E versus a F). While we perform this discrimination, the letters drift across the retina over distances much larger than their own sizes. While the brain can estimate motion using proprioceptive or efference copy signals, such information is not available at the required accuracy (refs?). Thus the eye's trajectory during drift must be estimated using the incoming spikes. almost copied from Burak et al intro, needs to be rewritten or cited properly

Burak and colleagues proposed a model that took an important first step in attempting to solve this problem from a probabilistic approach. They set up a probabilistic model and then use a form of approximate Bayesian inference. While this work was a great first step, it contains a number of assumptions that are unrealistic and that we have been able to relax. First, we allow for eye movements to take on continuous values. Second, we allow for pixels to be continuously valued (instead of binary). Third, we can incorporate prior knowledge about the distribution of images. Add refs for image priors. In order to make these generalizations, we do a point estimate of the image instead of estimating the posterior distribution of the image given the spikes as in the Burak model. **?** (also note a preceding paper **?**).

An important review of fixational eye movements as an information processing stage is given in **?**

Fig. 1 gives a rough overview of our approach. We begin with an image and generate spikes according to a fixational eye movement simulator. Then we decode those spikes, trying to recover the image and the trajectory of the image.

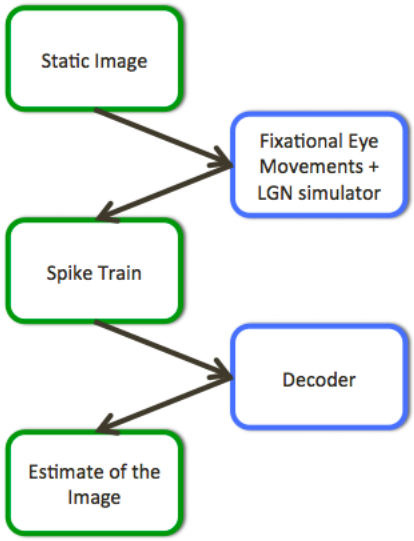

Figure 1: The goal of this project is to attempt to understand how V1 can estimate latent features of an image given the incoming signals from the LGN. Green boxes refer to data and blue boxes correspond to functions that transform the data. We start with a static image. Next, we run the image through a fixational eye-movement and LGN simulator to get spikes. We assume that the image is coded using a rate code and that diffusion occurs in continuous space. We use discrete time windows because it vastly speeds up the computation (in comparison to an event driven model). Next, we use a decoder (MAP estimate of a Bayesian model) to estimate a sparse code of the image given the spikes.

2 Model Specification

Begin with an image. For simulation purposes, we assume that fixational eye movements can be modeled by a discrete time, continuous space diffusion process. We assume that the LGN neurons have point-like receptive fields. Each neuron uses a rate code to communicate the extent to which its receptive field is activated by the image.

Here, we concretely specify the probabilistic graphical model that underlies this work. Define the following quantities:

1. S is the image. S_i denotes a particular pixel. We constrain S to be between 0 and 1.
2. A is the vector of sparse coefficients that generate S through a sparse coding dictionary, D. A_k denotes the kth sparse coefficient.
3. D is a sparse coding dictionary where D_k is the kth dictionary element.
4. X_t denotes the position of the center of the retina relative to the image at time t. Note that X_t is a vector with two components as the eye moves in two dimensions.
5. D_C is the diffusion coefficient of the eye movements, λ_0, λ_1 are the baseline and maximum firing rates of the neurons.
6. $R_{t,j}$ denotes number of spikes of LGN neuron j in the time window $[t, t + \Delta t]$. Δt is the timestep. We use R as an abbreviation for $R_{t,j}$ for all t and j.

For the rest of the paper, we will use the following subscripts consistently:

1. t: time step
2. b: batch
3. i: pixel
4. j: LGN neuron
5. k: sparse factor

These quantities are related through a probabilistic graphical model in Fig. 2. In order to specify a graphical model, we need to specify the probability of each node given the parents of that node. We systematically describe them below:

To begin with, we model the fixational eye movements as a simple diffusion process with a diffusion constant D_C (note we ignore the normalization constant in the probability):

$$p(X_0) = \delta(X_0) \tag{1}$$

$$-\log p(X_t|X_{t-1}) = \frac{1}{2(D_C/2)\Delta t}(X_t - X_{t-1})^2 \tag{2}$$

$$\tag{3}$$

Note that X_t is a two dimensional vector, so for the overall vector to have a diffusion constant of $D_C\Delta t$, then each individual component has a diffusion constant of $D_C/2\Delta t$. Next, we model the spiking of the neurons as a Poisson process. Let $R_{t,j}$ denote the number of spikes

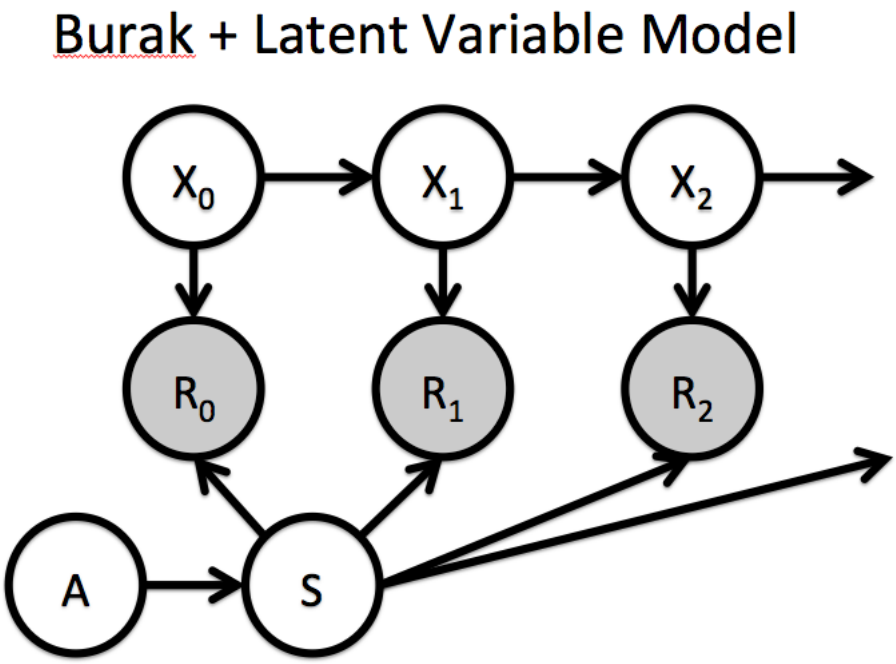

Figure 2: Here we see the structure of the probabilistic graphical model underlying this work. Notice that this is the Burak model with an additional node, A, which corresponds to latent features that generate the image, S.

emitted by neuron j at time t (specifically in the time interval $(t, t + \Delta t)$).

$$p(R_{t,j}|S, X_t) = \text{Poisson}(\lambda_j(S, X_t)) \tag{4}$$

$$\lambda_j(S, X_t) = \exp\left(\log \lambda_0 + \log(\lambda_1/\lambda_0) \cdot g \cdot S \cdot T_{X_t} E^j\right) \tag{5}$$

$$-\log p(R_{t,j}|S, X_t) = \delta_{R,1} \log \lambda_j(S, X_t)dt + \delta_{R,0} \log(1 - \lambda_j(S, X_t)dt) \tag{6}$$

where T_X denotes a translation by an amount X and E^j is the receptive field of RGC j. We model these receptive fields as a point at some location. g is a gain factor that sets the maximum size of $S \cdot T_{X_t} E^j$ to be approximately 1 (the $\cdot$ stands for a dot product, and this calculation is precisely defined later on). Note that we assume that $\lambda_1 \Delta t$ is much less than 1 so we approximate the poisson process as a Bernoulli process with $p = \lambda \Delta t$. In practice $p = 0.1$. Also note that other choices of the firing rate function are an option to be explored.

Finally, we have to specify a prior on our image. In this paper, we explore the advantage that we get when we incorporate a more substantial prior. As a baseline, we begin by assuming a simple independent pixel prior:

$$-\log p(S) = \sum_i -\log p(S_i) \tag{7}$$

$$-\log p(S_i) = \alpha * (S_i - 0.5)^2 + \beta * (\Theta(S_i - 1) + \Theta(-S_i)) \tag{8}$$

where $\Theta(x)$ is 1 if $x > 0$ and zero otherwise and parameters α and β are chosen using a bit of trial and error (insert how). Next, we describe a more complex prior that comes from the literature on natural scene statistics (insert refs):

$$-\log p(S|A) = \alpha \cdot (S - DA)^2 \tag{9}$$

$$-\log p(A) = \beta \sum_k |A_k| \tag{10}$$

Following along with the graphical model framework, if N are all the nodes of the graphical model, then

$$p(N) = \prod_i p(N_i|N_{\pi(i)}) \tag{11}$$

where $\pi(i)$ denote the parents of node i. Then all other quantities that we are interested in are specified by marginalizing the resulting distribution. In our case, we have

$$p(R, X, S) = \prod \text{factors in the list above} \tag{12}$$

Finally, one can use marginalization and Bayes rule to compute all other probabilities associated with the model (eg. $p(R) = \sum_{S,X} p(R, X, S)$).

3 Implementation Details

3.1 Determining Image - Receptive Field Overlap

In order to facilitate optimization of our model, we need to evaluate $S \cdot (T_{X^R} E^j)$. (Because we have many different positions to keep track of, denote the center of the retina (X^R, Y^R)). A reasonably easy way to implement this is to represent each image S as a set of gaussian bumps and model E^j as a point with center (X^E_j, X^E_j). Eg. the receptive fields of LGN cells are just pixels.

$$E_j(x, y) = \delta(x - X^E_j)\delta(y - Y^E_j) \tag{13}$$

where X^E, Y^E are the center of the receptive field. Then the translation operator is just adding $X^E \to X^E + X^R_t$. We represent an image as a collection gaussian bumps:

$$S(x, y) \to \sum_i S_i \frac{1}{2\pi\sigma^2} \exp\left[-\frac{(x - X^S_i)^2 + (y - Y^S_i)^2}{2\sigma^2}\right] \tag{14}$$

where $\sigma = 0.5$ is a uniform blurring factor and (X^S_i, Y^S_i) is the center of pixel i in our representation of the image. X^S, Y^S form a uniform grid with spacing 1. Then inner product of these two functions is

$$\int_{dx,dy} S(x, y) T_{X^R_t} E_j(x, y) = \sum_i \frac{S_i}{2\pi\sigma^2} \exp\left[-\frac{(X^S_i - X^E_j - X^R_t)^2 + (Y^S_i - Y^E_j - Y^R_t)^2}{2\sigma^2}\right] \tag{15}$$

While it is evident that this computation is wasteful because many terms in the sum are zero, we can easily parallelize the computation on GPU.

3.2 Kernel Trick to Reduce Memory

Suppose that we have some image $S_{u,v}$ (of shape L by L) where u, v denote the pixel index in the horizontal and vertical directions, respectively. Further, suppose we want to compute an inner product of the image with some collection of filters $A^j_{u,v}$ (where we have I total filters. The normal computation:

$$p^j = \sum_{u,v} I_{u,v} A^j_{u,v} \tag{16}$$

takes $|j| \cdot |u| \cdot |v| = J * L^2$ steps. Note that we need to have $J * L^2$ memory locations to save the filters $A^J_{u,v}$. This memory cost can be reduced.

However, suppose that our filters, $A^j_{u,v}$ have the additional structure that $A^j_{u,v} = B^j_u \cdot C^j_v$. I.e. the filter is an outer product of a horizontal part and a vertical part. The case relevant to our work is that if the $A^i_{u,v}$ are gaussians with a diagonal covariance matricies (they admit this decomposition). With that being done, we can rewrite

$$p^j = \sum_{u,v} I_{u,v} B^j_u C^j_v = \sum_v C^j_v \sum_u I_{u,v} B^j_u = \sum_v C^j_v \hat{I}^j_v \tag{17}$$

The computation of $\hat{I}_v^j$ takes JL^2 steps as before. Then the final sum takes JL steps. In terms of memory, both B and C need only JL spots in memory. As we can see, filters that have the form of an outer product can be optimized to use a factor of L less memory.

4 EM Algorithm

(Note in this section, we skip some details, to be explained more clearly in a later section).

Moving into the EM framework, let us treat R as an observed quantity, $X = X_{0:T}$ as an unobserved hidden variable and S as a parameter of the model. The EM algorithm maximizes a lower bound on the log posterior

$$\log p(S|R) = \log \sum_x p(R, X, S) - \log p(R) \tag{18}$$

(Note as R is fixed observed data, we can ignore the last term.) Generally, this type of calculation is intractable because we cannot calculate that sum for all values of S. Roughly speaking, EM replaces X by expected values of X given R, S (more precisely, we write out the complete log-posterior and we replace sufficient statistics of X with their expectations). Anyways, going along with the EM recipe, we write the full probability:

$$\log p(R, X, S) = \sum_{t,j} \log p(R_{t,j}|X_t, S) + \sum_t \log p(X_t|X_{t-1}) + \log p(S) \tag{19}$$

In the EM recipe (note we just follow the recipe here, and give the derivation in subsequent sections), we now assume that we begin with some estimate of say S, then estimate X, and alternate estimating these variables. So suppose that we have a good estimate of $S \to S'$. We take the expectation of the complete log-posterior defined below:

$$E_{p(X|R,S')} \log p(R, X, S) \tag{20}$$

As this equation can be a bit confusing, let me clarify a few things: R is fixed, and we choose a particular value of S', so this induces a distribution $p(X|R, S')$ on X. Then we take the expectation of the complete log-posterior where we deliberately note that we have S, not S' in $\log p(R, X, S)$. Thus our result is a function of S. In order to carry out this expectation, we take samples with associated weights

$$(X^m, W^m) \sim p(X|R, S') \qquad m = 1, \dots M \tag{21}$$

using a particle filter type algorithm. In particular, we note that we get the following estimate:

$$p(X|R, S') = \sum_m W^m \delta(X, X^m) \tag{22}$$

where W^m are weights and X^m are samples that correspond to that weight. Thus the ECLP (expected complete log probability) is approximated as:

$$\sum_{m=1}^{M} W^m \log p(R, X^m, S) \tag{23}$$

This is the so-called E-step of EM. Now for the M step. (See later for some more techincal details). We take the expectation that we just calculated and we maximize with respect to S. Note that the term corresponding to the diffusion of the position drops out as it does not depend on S. Thus we just need to maximize the expectation of the following:

$$\sum_{t,j} \log p(R_{t,j}|X_t, S) + \log p(S) \tag{24}$$

4.1 EM Details

We will give some more details about the EM recipe that we used in the previous section. The starting point is that we want to do MAP estimation of the posterior for the image given the observed spikes. We will instead look at a lower bound that has better computational properties. We begin with some fixed observed data, R. We want to find the following:

$$\hat{S} = \operatorname*{argmax}_S p(S|R) = \operatorname*{argmax}_S p(S, R) = \operatorname*{argmax}_S \log \sum_X p(S, X, R) \tag{25}$$

because $p(R)$ is a fixed number and log is an increasing function. As it stands, maximization is intractable. The idea now is to develop a lower bound of the function that can be maximized in a more tractable way. Introduce a variational distribution $q(X)$ subject to the constraint $\sum_X q(X) = 1$.

$$\log \sum_X p(S, X, R) \cdot \frac{q(X)}{q(X)} \geq \sum_X q(X) \log \frac{p(S, X, R)}{q(X)} \equiv \mathcal{L}(S, q) \tag{26}$$

for all $q(X)$ using Jensen's inequality as log is a concave function. $\mathcal{L}$ takes as inputs a function q and an image S. Crucially, the logarithm directly acts on $p(S, X, R)$ and splits that product into a sum. As we are interested in maximizing the LHS, we can try to maximize $\mathcal{L}$ instead. The idea of EM is to alternate maximizing this expression with respect to S and q. Specifically, suppose that we start with an estimate $S = S^0$, then we iterate the following recursions:

$$\text{Fix } S^t \tag{27}$$

$$q^t = \operatorname*{argmax}_q \mathcal{L}(S^t, q) \tag{28}$$

$$S^{t+1} = \operatorname*{argmax}_S \mathcal{L}(S, q^t) \tag{29}$$

The maximization with respect to S is a maximization in the typical sense and is called the M step:

$$\hat{S} = \operatorname*{argmax}_S \sum_X q(X) \log P(S, X, R) \tag{30}$$

where we note that the term $S[q] = -\sum_X q(X) \log q(X)$ does not depend on S. The maximization with respect to q is a bit different because q is a function, not a vector. We

can see that $\mathcal{L}$ is a KL divergence between $q(X)$ and $p(S, X, R)$, which is minimized when $q(X) = p(X|S, R)$ (since $q(X)$ is normalized). Note that here, we use the current estimate of S, which we call S'. So $q(X)$ at iteration t is $p(X|S', R)$. In summary, the steps of our algorithm are as follows:

1. Initialization: Pick a value of $S^t = S'$.

2. E step: Given a fixed value of S^t, maximize $\mathcal{L}(S^t, q)$ with respect to q. As shown above, this function is:
$$q^t(X) = p(X|S^t, R) \tag{31}$$
Note that we estimate this distribution using a particle filter as described in the section on particle filtering. Note that we don't actually compute $\mathcal{L}$. Also note that one can show that $\mathcal{L}(S^t, q^t) = \log p(S^t|R)$, so our lower bound is tight.

3. M step: find the argmax of $\mathcal{L}(S, q^t)$ with respect to S. Plugging in the value from the previous step gives us:
$$\hat{S}^{t+1} = \underset{S}{\operatorname{argmax}} \sum_X p(X|S^t, R) \log p(S, X, R) \tag{32}$$
Note that it is crucial at the value of S is left as a free variable in the log probability and that we use the previous value of $S = S^t$ to estimate the distribution for X. Note that we can disregard terms that do not depend on S, so we can rewrite the above as:
$$\hat{S}^{t+1} = \underset{S}{\operatorname{argmax}} \sum_X p(X|S^t, R)[\log p(R|S, X) + \log p(S)] \tag{33}$$

4.2 Particle Filtering

Assuming that we have a fixed image, $S = S'$, we can use a particle filter to obtain samples of the hidden state. Since we have a HMM, we can take advantage of the simplifications associated with such a model. Following the tutorial **?**, the simplest thing we can do is sequential importance sampling with resampling. (Eg. see section 4.1, note we make the replacements $1 : n \to 0 : t$, X remains the hidden state, Y becomes R, and S is given). To review the formulation in this paper, suppose that we have some sequence of distributions $(t = 0, \ldots T)$:
$$\pi(X_{0:t}) = \frac{\gamma(X_{0:t})}{Z_t} \tag{34}$$
where π is normalized but γ is not normalized. SMC methods help us sample from these distributions iteratively. We seek to estimate this distribution using a bunch of samples with weights:
$$\pi(X_{0:t}) \approx \sum_m W_t^m \delta(X_{0:t}, X_{0:t}^m) \tag{35}$$

The weights are defined by adding an auxillary distribution that is easier to sample from called an 'importance density,' or a proposal distribution, $q_t(x_{0:t})$ chosen by the user:

$$w_t(X_{0:t}) = \frac{\gamma_t(X_{0:t})}{q_t(X_{0:t})} \tag{36}$$

Note in order to have the computation per sampling step not increase linearly in time, we choose the particular form of the importance density:

$$q_n(X_{0:n}) = q_{n-1}(X_{0:n-1})q_n(X_n|X_{0:n-1}) \tag{37}$$

If we seek to minimize the variance of the estimator generated by our samples, we can show that the best choice is

$$q_t^{opt}(X_t|X_{0:t-1}) = \pi(X_t|X_{0:t-1}) \tag{38}$$

For our purposes, we are concerned with the special case where we are sampling from a Markov model:

$$\gamma(X_{0:t}) = p(X_{0:t}, R_{0:t}|S') = \prod_{n=0}^{t} g(R_n|S', X_n) \prod_{n=1}^{t} f(X_n|X_{n-1})\mu(X_0) \tag{39}$$

Then $\pi(X_{0:t}) = p(X_{0:t}|R_{0:t}, S')$ and $Z_t = p(R_{0:t}|S')$. Note that π is the distribution that we want to sample from to get the expectation for the EM solution. The variance optimal choice for the importance density, q_t^{opt}, then turns out to be:

$$\pi_t(X_t|X_{0:t-1}) = p(X_t|X_{0:t-1}, R_{0:t}, S') = p(X_t|X_{t-1}, R_t, S') \tag{40}$$

due to the conditional independencies that result from the HMM graphical model. The exact form of this distribution is difficult to sample from and it is reasonable to try to an approximation. In this paper, we use the proposal distribution:

$$q_t = p(X_t|X_{t-1}) \tag{41}$$

In practice, this estimated distribution seems to work well. Given this framework, there are a number of sampling techniques that we can use. The simplest one is the sequential importance resampling (SIR), that has the following steps:

1. Sample $X_t^i \sim q_t(X_t|Y_t, X_{t-1}^i)$
2. Compute the weights $W_t(X_t^i) \sim W_{t-1}(X_t^i)\frac{g(R_t|S', X_t^i)f(X_t^i|X_{t-1}^i)}{q_t(X_t^i|Y_t, X_{t-1}^i)}$
3. Resample if the effective sample size goes below threshold (eg. half of the number of particles).

Note that the ESS is defined as the reciprocal of the sum of the squares of the weights. If the weights are all equal, then the ESS is the number of particles. There is one important subtlety to note in this sampling process. At step t, W_t is the weight associated with $X_{0:t}$. That is to say, at each step, each particle represents a full path of X from 0 to t. Thus we get an approximation of $\pi(X_{0:t})$. However, due to the choice of the sampling distribution to go from t to $t+1$, we just copy over $X_{0:t}$ and sample the last component $X_{t+1} \sim q(X_{t+1}|X_{0:t}) = q(X_{t+1}|X_t)$. Thus we need to only store the particle at each time point to get the particles at the next time point.

Some other details that are useful for implementation are that one can do resampling in $O(|i|)$ steps using a method called systematic resampling.

There is an important subtlety associated with the resampling. As formulated above, the size of the samples grows. Eg. after the tth step, the particle corresponds to the full path $X_{0:t}$. At the end of the algorithm, we have particles that sample the full path $X_{0:T}$. However, due to resampling, the estimates for early times are going to be mostly the same and the estimate is going to be poor. Instead, it works better in practice to estimate X_t by looking at the marginal of the particles from step t, eg. $X_{0:t}$. This can be written more formally as follows: We want:

$$E_{p(X_{0:T}|R_{0:T},S')} \sum_t \log p(R_t|X_t, S) = \sum_t \int dX_t \, p(X_t|R_{0:T}, S') \log p(R_t|X_t, S) \tag{42}$$

where we got rid of the integration over all X except X_t. So how do we estimate these marginals? It is sufficient to estimate these using the filtering estimate:

$$p(X_t|R_{0:T}, S') \approx p(X_t|R_{0:t}, S') \tag{43}$$

In principle, it is possible to do particle smoothing to estimate $p(X_t|R_{0,T}, S)$, but that is unnecessarily complicated. Also as we are seeking to move towards a causal algorithm, we prefer to use the filtering solution. Thus our full equation is as follows:

Define W_t^m to be the weight of $X_{0:t}$ for the mth sample. Define X_t^m to be the value of X_t from the mth particle filter sample at time t (i.e. extract out the last time point from $X_{0:t}^m$) using a particle filter on the HMM corresponding to $p(X|R, S^t)$. [1] Thus we get:

$$S^{t+1} = \operatorname*{argmax}_S \left[\sum_m \sum_t W_t^m \log p(R_t|X_t^m, S) + \log p(S) \right] \tag{44}$$

4.3 Creating images showing EM estimate improving over time

In my presentations on this topic, I show a video that shows the estimate of the image improving over time. The basic idea for this is as follows: I use EM to estimate S, X given spikes from a time interval $[0, T_1]$. Eg. if the total time interval of generated spikes is $T = 200ms$, then T_1 might be $10ms$. Here, we get an estimate S^1 of the image. Then the

[1] This is in contrast to using W_T^m and X_t^m from $X_{0:T}^m$.

next step is that we run EM where on the first $T_2 = 20ms$ of data starting with the E step and then the M step. This gives us $\hat{S}^2$, and so on. Roughly speaking S^k is the approximate solution to $\text{argmax}_S\, p(S|R_{0:T_k})$.

4.4 Optimizing the Objective Function with respect to the image

While naive gradient descent works well in theory, it doesn't work so well in practice. As we have noted, the choice of the firing rate function $\lambda(S)$ gives us a convex optimization problem. While we could use a second order method, it is easier to program a quasi-second order method (although it might still be worth trying a fully second order method). In this work, we use ADADELTA. The goal of this first order method is to use running RMS averages of the gradients and parameter changes to divide out constants in the function. Eg. we should be invariant to $E(x) \to a \cdot E(b \cdot x)$. Suppose the function that we are trying to minimize is $E(x)$ and we get a gradient $g = \frac{\partial E}{\partial x}$.

1. Fix constants ρ, ϵ, Initialize auxillary variables $Edx2 = 0$, $Eg2 = 0$.
2. Calculate $g = \frac{\partial E}{\partial x}$.
3. Update $Eg2 = \rho * g + (1 - \rho)Eg2$
4. Calculate $dx = \frac{\sqrt{\epsilon + Edx2}}{\sqrt{\epsilon + Eg2}} * g$
5. Update $Edx2 = \rho * dx + (1 - \rho)Edx2$.
6. Update $x \to x - dx$.
7. Repeat.

Note that these calculations are done elementwise. Typical parameter values are $\rho = 0.9$ and $\epsilon = 0.001$. In the expression for dx, we see that the units of E and x should cancel out.

4.5 Optimizing the Objective Function

Since the objective function has an absolute value, typical gradient descent approaches converge slowly. Thus there are special purpose gradient descent methods that minimize functions that are in the form

$$f(x) + g(x)$$

where $f(x)$ is a continuously differentiable, convex function and $g(x)$ is a convex, but not continuously differentiable function, such as $g(x) = \alpha|x|$. One such method is called FISTA, or the Fast Iterative Shrinkage-Threshold Algorithm. The core kernel of the FISTA algorithm is the ISTA step. Define

$$p_L(y) = \text{argmin}_x\, g(x) + L/2 * ||x - g(y)||^2$$

where

$$g(y) = y - \frac{1}{L}\nabla f(y)$$

and where L is the constant such that

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$$

When $g(x) = \alpha|x|_1$, then

$$p_L(y) = h_\theta(g(y)) \qquad h_\theta(u) = \text{sign}(u)(|u| - \theta) \qquad \theta = \frac{\alpha}{L}$$

h is applied pointwise to u and is the shrinkage function. Simplying calculating $x_{t+1} = p_L(x_t)$ is the ISTA algorithm. If we more intelligently choose our new value to probe our function, then we get faster convergence. The FISTA algorithm is as follows:

1. Initialize $y_0 = x_0 = X0$, $t_0 = 1$.
2. For $k \geq 0$, iterate the following:

$$x_{k+1} = p_L(y_k) \qquad t_{k+1} = 0.5 * (1 + \sqrt{1 + 4 * t_k^2}) \qquad y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}} * (x_{k+1} - x_k)$$

4.6 Choosing the Lipschitz Constant

Our objective function is:

$$E(A) = \sum_{b,j,t} W_t^b h_{t,j} \left(\sum_{i,k} D_{i,k} A_k K_{b,t,i,j} \right) \tag{45}$$

where $K_{b,t,i,j} = g * \exp(-(X_i^S - X_{b,t}^R - X_j^E)^2/\sigma^2)$ is the link between pixels and receptive fields of the neurons and $h_{t,j}(u) = R_{t,j} \log f(u) - f(u)dt$, $f(u) = \exp(\log \lambda_0 + \log \frac{\lambda_1}{\lambda_0} u)$, D is the dictionary, g is the gain factor. Note that g is chosen so that $\sum_i K_{b,t,i,j} \leq 1$. We also normalize our dictionary so that $\sum_i D_{i,k}^2 = 1$. Now we have to find the Lipschitz constant of this function. First, when calculating this derivative, we have

$$\frac{\partial E}{\partial A_k} = \sum_{b,j,t} W_t^b \cdot h'_{t,j}(\sum_{i,k} D_{i,k} A_k K_{b,t,i,j}) \sum_i D_{i,k} K_{b,t,i,j} \tag{46}$$

We want to compute the difference of this derivative for two different vectors $A = U, V$. The first term of the derivative drops out because it is the same for U, V, then we take the difference of the following for $A = U, V$:

$$\sum_{b,j,t} W_t^b dt \log \frac{\lambda_1}{\lambda_0} \cdot f(\sum_{i,k} D_{i,k} A_k K_{b,t,i,j}) \sum_i D_{i,k} K_{b,t,i,j} \tag{47}$$

Since $|\sum_i w_i x_i| \leq \sum w_i |x_i| \leq L$ if $|x_i| \leq L$ and w_i are weights, it suffices to look for an upper bound for each batch and time point independently.

I.e. our upper bound is $dt \log \frac{\lambda_1}{\lambda_0} * N_T * N_N$ times the Lipschitz bound on:

$$f(\sum_{i,k} D_{i,k} A_k K_{b,t,i,j}) \sum_i D_{i,k} K_{b,t,i,j} \tag{48}$$

To form an upper bound, we note that the mean value theorem that there is a z such that $|f(x) - f(y)| \leq |x - y| f'(z)$ for all $x < z < y$. So we need to find the upper bound on the derivative of the function above. This gives us:

$$\log \frac{\lambda_1}{\lambda_0} \cdot f(\sum_{i,k} D_{i,k} A_k K_{b,t,i,j})(\sum_i D_{i,k} K_{b,t,i,j})^2 \tag{49}$$

So for this function, we have an additional constraint that the firing rate has a maximum value of λ_1, so that bounds f. Also we can see that $\sum_i D_{i,k} K_{b,t,i,j} \leq 1$ because of the size of each. Finally, since we do the minimization independently for each k, we have our bound:

$$\lambda_1 dt \left(\log \frac{\lambda_1}{\lambda_0} \right)^2 * N_t * N_N \tag{50}$$

5 Discrete Space EM Algorithm in Real-time

A crucial issue with the previous work is that it doesn't translate well into a real time algorithm. The problem is that the sum of the likelihood functions for the image don't have a simple representation as they are a sum of the logarithm of poissons that depend on a mixture of the pixels (because we interpolate between pixels). If we break the assumption of continuous eye movements, we can

Making certain assumptions, we can derive an approximate inference algorithm that operates in real time. The fundamental trick is that we choose $\lambda(S)$ such that $\log \lambda(S)$ is linear in S. We start with deriving the E step: Let us begin by simplifying the follwing term (ignore $\log R_{t,j}!$ as it is constant):

$$\log p(R_{t,j}|X_t, S) = R_{t,j} \log \lambda(S \cdot T_{X_t} E^j) - \lambda(S \cdot T_{X_t} E^j) \tag{51}$$

$$= R_{t,j} \left(\log \lambda_0 + \log \frac{\lambda_1}{\lambda_0} S \cdot T_{X_t} E^j \right) - \exp \left(\log \lambda_0 + \log \frac{\lambda_1}{\lambda_0} S \cdot T_{X_t} E^j \right) \tag{52}$$

Crucially, we see that the first term is linear in S so when we sum contributions from different terms, we can just keep track of the linear coefficients. Now assume that we have estimated $p(X_t|R_{0:t}) \equiv p_X^t(X_t)$ for $t = 0, 1, ..., T$. As in the EM recipe, we estimate S by taking the argmax with respect to S of the following expectation:

$$E_X \left(\sum_{t,j} \log p(R_{t,j}|X_t, S) \right) = \sum_{t,j} \sum_X p_X^t(X) \log p(R_{t,j}|X, S) \tag{53}$$

Now we calculate the expectation for the two terms in the equations above separately:

$$\sum_{t,j} \sum_X p_X^t(X) R_{t,j} \left(\log \lambda_0 + \log \frac{\lambda_1}{\lambda_0} S \cdot T_X E^j \right) \tag{54}$$

$$= \log \lambda_0 \sum_{t,j} R_{t,j} + \log \frac{\lambda_1}{\lambda_0} \cdot S \cdot \sum_{t,j,X} R_{t,j} p_X^t(X) T_X E^j \tag{55}$$

$$= k \cdot S \cdot \sum_t \sum_X p_X^t(X) \sum_j R_{t,j} T_X E^j \tag{56}$$

$$= k \sum_{X'} S(X') \sum_t \sum_X p_X^t(X) \sum_j R_{t,j} \delta(X', X + X_j^E) \tag{57}$$

$$= k \sum_{X'} S(X') \sum_{t,j} p_X^t(X' - X_j^E) R_{t,j} \tag{58}$$

where we ignore the first term because it does not depend on S. Essentially what we do here is that we use $p_X^t(X)$ to route the spikes that come in at time t to the corresponding pixel. The probability calculations give us the appropriate way to weight this information. Also note that if $T_X E^j$ is out of bounds of the image, we set that dot-product to zero. Now we treat the negative phase term (note that this does not depend on the data $R_{t,j}$):

$$\sum_{t,j}\sum_{X} p_X^t(X)\exp\left(\log\lambda_0 + \log\frac{\lambda_1}{\lambda_0} S\cdot T_X E^j\right) \tag{59}$$

$$= \lambda_0 \sum_{t,j,X} p_X^t(X)\exp\left(k\cdot S(X+X_j^E)\right) \tag{60}$$

$$= \lambda_0 \sum_{X'} \exp(k\cdot S_{X'}) \sum_{t,j} p_X^t(X'-X_j^E) \tag{61}$$

where we use the fact that each receptive field is just a pixel and $S(X) = S_X$ denote the pixel at location X. We also note that if $X + X_j^E$ is out of bounds of the image, then the dot product $S\cdot T_X E^j$ is zero and $\exp(0) = 1$. The resulting terms $\sum p_X^t$ are constants that do not depend on any of the pixels, so we ignore them. Also $k = \log\frac{\lambda_1}{\lambda_0}$.

Thus we see that in order to perform the maximization with respect to S after T time steps, we just need to keep track of matrices:

$$\hat{S}^T = \underset{S}{\operatorname{argmax}}\left[\sum_{X'} kS_{X'}A^t(X') - \lambda_0\exp(k\cdot S_{X'})B^t(X') + \log p(S)\right] \tag{62}$$

$$A^t(X') = \sum_t\sum_j p_X^t(X'-X_j^E)R_{t,j} \tag{63}$$

$$B^t(X') = \sum_t\sum_j p_X^t(X'-X_j^E) \tag{64}$$

Roughly speaking, $A(X')$ keeps track of the dynamically routed data for each pixel X' and $B(X')$ keeps track of how often pixel X' has been in our field of view.

In order to do the E step, we simply use our estimate for S and use Bayes rule:

$$p_X^{T+1}(X) \sim p(R_{T+1}|X,\hat{S}^T)\cdot\sum_{X_T} p(X|X_T)p_X^T(X_T) \tag{65}$$

In otherwords, we fix S to be our estimate from the previous time step, then use the HMM rules to update our position estimate for the current time point. We can summarize the algorithm in the following steps:

1. Initialization: $A^{-1}(X) = B^{-1}(X) = 0$. $p_X^0(X) = \delta(X,0)$.

2. For $t = 0, 1, \ldots$.:

$$A^t(X) = A^{t-1}(X) + \sum_j p_X^t(X - X_j^E) R_{t,j} \tag{66}$$

$$B^t(X) = B^{t-1}(X) + \sum_j p_X^t(X - X_j^E) \tag{67}$$

$$\hat{S}^t = \operatorname{argmax}_S \sum_X \left(k S_X A^t(X) - \lambda_0 \exp(k S_X) B^t(X) \right) + \log p(S) \tag{68}$$

$$p_X^{t+1}(X_{t+1}) \sim p(R_{T+1} | X_{t+1}, \hat{S}^t) \cdot \sum_{X_t} p(X_{t+1} | X_t) p_X^t(X_t) \tag{69}$$

Note that X here corresponds to the location of a particular pixel. Roughly speaking, these quantities are correspond to

$$\hat{S}^t \approx \operatorname*{argmax}_S p(S | R_{0:t}) \tag{70}$$

$$p_X^t(X_t) \approx p(X_t | R_{0:t}) \tag{71}$$

A gives us the dynamically routed information from the incoming spikes and B gives us the expected number of timesteps that a particular pixel was in view of the retina.

6 Online EM Using Gaussian Estimation

With a further approximation, we can create an online algorithm for solving our problem. The basic idea is to do a series expansion of the image likelihood $\log p(R|X,S)$ about the current estimate of S. Then as a sum of quadratics is a quadratic, we can do the inference for S using an amount of memory that is fixed in time. It should be noted that while this approximation leads to theoretically nice estimation, the quality of the performance has yet to be demonstrated. We can build up our estimate recursively. Suppose that for each time point, we have an estimate of the position $p_X^t(X_t)$ as in the previous section. Using the EM idea, we say that

$$\hat{S}^t = \underset{S}{\operatorname{argmax}} \sum_X \sum_{t'=0}^{t} \log p(R_{t'}|X_{t'},S) p_X^{t'}(X_{t'}) \tag{72}$$

(This isn't strictly EM because we are approximating the distribution $p(X_{0:t}|R_{0:t},S)$ with marginals derived from each time step). Now we want an approximation to this cost function that allows for online inference. A simple idea is to use a second order taylor expansion about the current estimate of the image. Eg. make the replacement

$$\log p(R_t|X_t,S) \to c^t(X_t) + B^t(X_t)\cdot(S-\hat{S}^t) + \frac{1}{2}(S-\hat{S}^t)^T \cdot M^t(X_t)\cdot(S-\hat{S}^t) \tag{73}$$

$$c^t(X_t) = \log p(R_t|X_t, S=\hat{S}^t) \tag{74}$$

$$B^t(X_t) = \frac{\partial}{\partial S}\log p(R_t|X_t,S)|_{S=\hat{S}^t} \tag{75}$$

$$M^t(X_t) = \frac{\partial^2}{\partial^2 S}\log p(R_t|X_t,S)|_{S=\hat{S}^t} \tag{76}$$

Note that c_t doesn't depend on S, so we ignore it. B^t is a vector, and M^t is the hessian matrix. Plugging these into the EM equation gives us:

$$\hat{S}^t = \underset{S}{\operatorname{argmax}} \sum_X \log p(R_t|X_t,S) p_X^t(X) + \sum_{t'=0}^{t-1} p_X^{t'}(X_{t'})\cdot\left(B^{t'}(X_{t'})(S-\hat{S}^{t'}) + \frac{1}{2}\cdot(S-\hat{S}^{t'})^T M^{t'}(X_{t'})(S-\hat{S}^{t'})\right) \tag{77}$$

Note we haven't yet estimated $\hat{S}^t$, so we leave the $\log p(R_t|X_t,S)$ term alone. Now we can take the expectation over X, and we get that the second term is

$$\sum_{t'=0}^{t-1} B^{t'}(S-\hat{S}^{t'}) + \frac{1}{2}(S-\hat{S}^{t'})^T M^{t'}(S-\hat{S}^{t'}) \tag{78}$$

where we make the abbreviations

$$M^t = \sum_{X_t} p_X^t(X_t) M^t(X_t) \tag{79}$$

$$B^t = \sum_{X_t} p_X^t(X_t) B^t(X_t) \tag{80}$$

$$= \frac{1}{2} S^T \left(\sum_{t'=0}^{t-1} A^{t'} \right) S + S^T \cdot \left(- \sum_{t'=0}^{t-1} M^{t'} \hat{S}^{t'} + B^{t'} \right) \tag{81}$$

Going back to our original expression, we have

$$\hat{S}^t = \underset{S}{\operatorname{argmax}} \left[\log p(S) + \frac{1}{2} S^T \tilde{M}^{t-1} S + S^T \tilde{B}^{t-1} + \sum_{X_t} p_X^t(X_t) \log p(R_t | X_t, S) \right] \tag{82}$$

Thus we see that the previous observations are combined into a simple quadratic form. Now lets look again at the derivatives that we need to take. First we write out the likelihood that we are differentiating:

$$\log p(R_t | X_t, S) = \sum_j \log p(R_{t,j} | X_t, S) = \sum_j g_{t,j}(S \cdot T_{X_t} E^j) \tag{83}$$

where $g_{t,j}(x) = R_{t,j} \log \lambda(x) - \lambda(x)$. Then the first derivative is:

$$\sum_j g'_{t,j}(...) T_{X_t} E^j \tag{84}$$

where we note that $T_{X_t} E^j$ is a vector. Further, the second derivative is:

$$\sum_j g''_{t,j}(...) (T_{X_t} E^j) \otimes (T_{X_t} E^j) \tag{85}$$

where $\otimes$ denotes the outer product between two vectors.

Let us summarize all of our steps:

1. Initialize $\hat{S}^{-1} = \operatorname{argmax}_S \log p(S)$
2. Initialize the matrix $\tilde{M}^{-1} = 0^{n \times n}$, and vector $\tilde{B}^{-1} = 0^n$.
3. Initialize $p_X^0(X_0)$ using the particle filter with $S = \hat{S}^{-1}$. (i.e. using the initial proposal distribution and reweighting using the likelihood $p(R_0 | X_0, S = \hat{S}^{-1})$).
4. Initialize $t = 0$

5. Calculate:

$$\hat{S}^t = \underset{S}{\operatorname{argmax}} \left[\log p(S) + \frac{1}{2} S^T \tilde{M}^{t-1} S + S^T \tilde{B}^{t-1} + \sum_{X_t} p_X^t(X_t) \log p(R_t | X_t, S) \right] \quad (86)$$

6. Calculate

$$B^t = \sum_{X_t} p_X^t(X_t) B^t(X_t) = \sum_{X_t} p_X^t(X_t) \frac{\partial}{\partial S} \log p(R_t | X_t, S)|_{S=\hat{S}^t} \quad (87)$$

$$M^t = \sum_{X_t} p_X^t(X_t) M^t(X_t) = \sum_{X_t} p_X^t(X_t) \frac{\partial^2}{\partial^2 S} \log p(R_t | X_t, S)|_{S=\hat{S}^t} \quad (88)$$

7. Calculate

$$\tilde{M}^t = \tilde{M}^{t-1} + M^t \qquad \tilde{B}^t = \tilde{B}^{t-1} + B^t - M^t \hat{S}^t \quad (89)$$

8. Calculate

$$p_X^{t+1}(X_{t+1}) \sim p(R_{t+1} | X_{t+1}, S = \hat{S}^t) \sum_{X_t} p(X_{t+1} | X_t) p_X^t(X_t) \quad (90)$$

   Note that we do this calculation using a particle filter with resampling.

9. Set $t = t + 1$ and return to step 5.

6.1 Justification of the Gaussian Approximation

When doing this type of gaussian approximation, it is worthwhile to think about how well we expect the algorithm to work. For a simple illustration, suppose that we are in the situation where the spacing of the pixels in the image and the spacing of the receptive fields are the same. Roughly speaking, our sum will then decompose into functions of individual pixels in the image instead of having cross terms. One such term in the objective function will be:

$$R_{j,t} \log f(s) - f(s) \quad (91)$$

where s is an individual pixel of the image. Now, let f taken on the value that we are using for this investigation, and we get:

$$R_{j,t} \left(\log \lambda_0 + \log \frac{\lambda_1}{\lambda_0} s \right) - \exp \left(\log \lambda_0 + \log \frac{\lambda_1}{\lambda_0} s \right) dt \quad (92)$$

Next, suppose that we expand this function out in terms of $s = \hat{s}$. Let $\hat{\lambda} = f(\hat{s})$. Ignoring the constant term, we get

$$[s - \hat{s}] \log \frac{\lambda_1}{\lambda_0} \left(R_{j,t} - \hat{\lambda} dt \right) - \frac{1}{2} [s - \hat{s}]^2 \left(\log \frac{\lambda_1}{\lambda_0} \right)^2 \hat{\lambda} dt \quad (93)$$

7 Online EM with Sparse Prior and Gaussian Estimation

Given all of the steps in the previous sections, we can more concisely articulate the method that we are doing in this section. In this section, we assume that $p(S|A) = \delta(S - DA)$. That is to say the image is exactly equal to

Motivated by the idea of EM, we estimate the argmax of the posterior of the sparse coefficients given the spikes. Since we cannot compute this exactly, we use the EM idea to approximate this posterior.

$$\hat{A} = \operatorname*{argmax}_A p(A|R) = \operatorname*{argmax}_A \sum_{S,X} p(A, S, X|R) = \operatorname*{argmax}_A \log \sum_{S,X} p(A, S, X, R) \tag{94}$$

Since $p(S|A)$ is a delta function, we can evaluate the sum over S straightforwardly:

$$\log \sum_{S,X} p(R|X, S)p(X)p(S|A)p(A) = \log \sum_X p(R|X, S = DA)p(X)p(A) \tag{95}$$

Using the EM idea, we fix an estimate for $A = A'$, and then we replace the sum over X with a variational distribution $q(X) = p(X|R, A = A')$. (Skipping a few steps, noting $\log p(X)$ doesn't depend on A.).

$$\hat{A} = \operatorname*{argmax}_A \sum_X q(X) \log p(R|X, S = DA) + \log p(A) \tag{96}$$

Now, we have not taken time into account. As we are modeling a biological system, we want to have an online algorithm. To go this direction, notice that the logarithm splits up the evidence coming in at different times and we only need the marginals of $q(X)$ to estimate each of these terms:

$$\operatorname*{argmax}_A \sum_X \sum_t q(X) \log p(R_t|X_t, S = DA) = \operatorname*{argmax}_A \sum_t \sum_{X_t} q(X_t) \log p(R_t|X_t, S = DA) \tag{97}$$

where $q(X_t)$ is the X_t marginal of $q(X)$ (recall $X = (X_0, \dots X_T)$). The EM idea suggests that at each iteration, we update the entire distribution $q(X)$, however, we can weaken this assumption and retain our estimates for past time points and only update for the most recent timestep. In addition to that, we expand out the terms using a Taylor series. This allows us to compute the sum in an online manner.

8 Image Priors: Moving Beyond the Independent Pixel Assumption

As discussed earlier, the whole reason of moving into this framework is that it lets us incorporate more general priors on the image (at the expense of treating the image with a point estimate instead of a distribution). As a first step, we use the simple prior:

$$-\log p(S) = \sum_i -\log p(S_i) = \alpha * (S_i - 0.5)^2 + \beta * (\Theta(S_i - 1) + \Theta(-S_i)) \tag{98}$$

where $\Theta(x) = x$ for $x \geq 0$, and zero otherwise. β is chosen to be sufficiently large to keep the pixels in the domain $[0, 1]$. So far, the results with this prior have been pretty good. Our framework naturally allows us to incorporate a more complex prior. As discussed above, we make the replacement:

$$-\log p(S) \rightarrow -\log p(S, A) = \alpha(S - DA)^2 + \beta \sum_j |A_j| \tag{99}$$

And correspondingly, we replace

$$\underset{S}{\text{argmax}} \rightarrow \underset{S,A}{\text{argmax}} \qquad p(S|R) \rightarrow p(S, A|R) \tag{100}$$

It would be nice to be able to have $p(S) = \sum_A p(S, A)$ as our prior but that calculation is not tractable. In the following work, we simply estimate A and S by doing alternating gradient steps for A and S.

8.1 MNIST Prior

Upon looking at the images produced with the image prior that is factorial over pixels, we see that the images less smooth than they should be. That is to say the random nature of Poisson neurons leads to some pixels having a couple spikes, whose effects have yet to be averaged out. An image prior helps us get past this problem. To illustrate this point, we will show the effects of adding a sparse coding prior to the image. As an illustrative example, we train a sparse coding dictionary on MNIST. To begin with, we took MNIST, and reduced the size of the images by half so they are 14 by 14 images. When doing sparse coding, we look at $(S - DA)^2 + k|A|$ where S is the image, D is a dictionary, and A is a set of sparse generative factors. k is a constant that changes the sparsity of the solution. For our purposes, we chose a value of k so that we are in a regime where the relative sizes of the reconstruction error and the sparsity penalty are the same order of magnitude. This ensures that the minimization procedure attempts to seek a trade off between sparsity and reconstruction error (instead of the limiting behaviors where one of those constraints are neglected). Thus we inherit a particular value of k for the purposes of the algorithm of thus paper.

Our objective function is roughly

$$-E_X \log P(R|S, X) + \alpha \left[(S - DA)^2 + k|A|\right] \tag{101}$$

While we don't have to deal with this problem in sparse coding, we notice that there is an additional parameter associated with the prior. We define α as the prior strength. We have to choose a value for this parameter. Note that the poisson spikes have no free parameter to choose and we are trying to use a physiologically reasonable value of the firing rate. The simplest way to look at the impact of α is to change its value and see what happens to the SNR as a function of time. As in the case of sparse coding, we see that there different qualitative regimes for the parameter α.

1. Degenerate: $\alpha = 0$ - when $\alpha = 0$, there is no gradient for A when minimizing the objective function.

2. Independent minimization: $\alpha << 1$ - here α is so small that it is as if we first minimize for S and then minimize for A. Eg. when we take the gradient with respect to S, we find the S that is most consistent with the observed data $\log p(R|S, X)$. Since α is small, the prior basically doesn't impact the minimization for S. With a value of S chosen, the gradient with respect to A chooses an A that is most consistent with the observed spikes.

3. Unstable: $\alpha >> 1$ - here α is so large that the prior dominates so that S is always approximately equal to DA, and the gradient due to the data doesn't have much impact until S and DA are very close. Here, the minimization seems unstable, although it seems like it it should be the case that things don't fall apart completely.

4. Reciprocal minimization: $\alpha = O(1)$. Here, we have the prior and the data make comparable contributions to the estimation of the image and the latent factors. Eg. When estimating S, the prior now regularizes the estimate of S. This serves the additional advantage of getting a better estimate of the image, which results in better position estimates.

In order to look at the performance of the algorithm, we consider the following: the estimated image $\hat{S}$ and the sparse reconstruction $D\hat{A}$. We can look at the SNR of the reconstruction between the actual image and these two estimates for the image. One property that might be surprising is that for certain regimes, the estimate DA is better than the estimate for S.

One interesting qualitative property is that in the right regime, it is easy to identify the correct number for a very short observation time (eg. $15ms$).

8.2 Natural Image Prior and Perceptually Relevant Cues

A next step in the story of this paper is to look at a natural image prior. A crucial issue to grapple with is that the brain isn't interested in a pixel by pixel reconstruction of the incoming images. We are looking for perceptually relevant cues from the images and looking to code them. Including sparse coding is a first step in this direction. Thus we are more interested in the sparse coefficients, A, rather than the image, S. S just serves as an intermediate to

estimate A. It is important here to think about the interpretation of S, and A, and the so noise in $p(S|A)$. Think about how to explain mismatch between image and latent factors

9 Motion Prior: Including Momentum and Acceleration

As we are beginning a collaboration with the Roorda lab that uses adaptive optics to track the motion of eye movements, it is evident that drift is not purely diffusive motion. Indeed, it seems to be the case that the eye motions seem to have some momentum to them. In this section, we describe how such models can be incorporated into the existing framework and how to use experimental data to fit these models.

9.1 Why include a motion prior?

Stepping back a bit, what is our goal in modeling motion from the perspective of the brain? Without any incoming sensory data, our estimate of the current position of the eye gets worse as time progresses. In the naive model, we assume that motion follows diffusion. In this model, if we know the position to be x_0 and a time dt passes, then the position estimate is a gaussian with center x_0 and a variance $\sigma^2 = Ddt$. When sensory input comes in, we reduce the variance of our estimate. In other words, there is a constant push and pull between increased uncertainty due to time and decreased uncertainty due to observations. However, if there is more structure to the motion than implied by the simple diffusion model, we can reduce 'push' that continually tries to increase our uncertainty. In other words, we want a model such that $p(X_{t+1}|X_t)$ has a smaller uncertainty.

To illustrate the usefulness of a motion prior, suppose that all of the motions that we were looking at had a constant velocity. In the naive view, we could still approach this problem using a diffusive motion prior. In this model, we would continually estimate the position, diffuse that estimate outwards, only to find that the object always ended up landing in the same direction relative to the center our estimate of the position. On the other hand, if we knew that the motion had a constant velocity, we use the first few observations to estimate the velocity. Then we use that velocity estimate to create a lower uncertainty estimate for the position prior to any observations for that time point. After getting a good estimate of the velocity, then we will always know the position with high certainty in this extreme example. Indeed, the goal of the investigation of this section is to see if adding a prior on the motion is helpful in recovering the image from the spikes.

9.2 A simple model of motion that can be integrated into the current framework

Suppose that X_t and V_t describe the actual position and velocity of the particle at time t. (Note X_t is a vector giving the horizontal and vertical position of the eye, similarly for V_t). Finally, suppose that $\hat{X}_t$ is our observed position. Notice that in this framework, it is important to distinguish between the actual position and the observed position. Assuming that there is zero acceleration during drift[2], we write out the following equations:

[2]the model can easily be adapted to nonzero acceleration

$$X_{t+1} = X_t + \Delta t * V_t \tag{102}$$
$$V_{t+1} = V_t + \epsilon_v \tag{103}$$

where Δt is the time interval between observations, ϵ_v is some Gaussian noise. Finally, our observations are described as measuring our position subject to some noise:

$$\hat{X}_t = X_t + \epsilon_{obs} \tag{104}$$

where ϵ_{obs} is some Gaussian noise.

Given this generative description of our observations, we can rephrase the problem in the language of a hidden Markov model (and since we are assuming everything is gaussian, we can solve the estimation of the hidden state given the observed state using a Kalman Filter or a Kalman Smoother). Finally, the estimation for X, V given $\hat{X}$ can be done using well-established techniques.

A hidden markov model has a hidden state, H_t, and an observed state, O_t. In our case, the hidden state is

$$H_t = \begin{bmatrix} X_t \\ V_t \end{bmatrix} \qquad O_t = \begin{bmatrix} \hat{X}_t \end{bmatrix} \tag{105}$$

In this vectorized notation, we then have:

$$H_{t+1} = \begin{bmatrix} I_2 & I_2 \cdot dt \\ 0_2 & I_2 \end{bmatrix} H_t + \epsilon_h \qquad O_t = \begin{bmatrix} I_2 & 0_2 \end{bmatrix} H_t + \epsilon_{obs} \tag{106}$$

where I_2 denotes the 2 by 2 identity matrix and 0_2 denotes the 2 by 2 zero matrix and ϵ_h and ϵ_o are gaussian noise that is uncorrelated in time. In order to fully specify the model, we need to specify the following things: (1) the mean and covariance matrix of the prior for the hidden state, (2) the covariance matrices for ϵ_h and ϵ_{obs}.

1. Hidden state prior $p(H_0)$: we just estimate that the mean estimate for the position is zero and the mean velocity is also zero, covariance matrix - assume that it is diagonal with zero uncertainty for x_0 and some parameter σ_p for the variance of the velocity estimate (this is one parameter needed to be fit).

2. Hidden state transition noise, ϵ_h: A simple assumption used in these applications is to assume a diagonal covariance matrix with nonzero noise for the velocity. This gives the second parameter σ_t. This gives us the hidden state transition probability $p(H_{t+1}|H_t)$.

3. Observation noise, ϵ_{obs}: The covariance matrix associated with the observations can also be reasonably assumed to be multiple of the identity and that gives us a us a third parameter σ_{obs} to estimate from the data. This gives us the observation probability $p(O_t|H_t)$.

Once we fix the parameters of the model, we can estimate the position and velocity from the observed position. Eg. we can use the Kalman filter to estimate $p(H_t|O_{0:t})$ or a Kalman smoother to estimate $p(H_t|O_{0:T})$ where T is the total number of timesteps of observation.

Note that if we want to have some sort of acceleration, then we just add that in as a variable, and we have $X_{t+1} = X_t + V_t\Delta t + \frac{1}{2} \cdot A_t\Delta t^2$, $V_{t+1} = V_t + A_t\Delta t$, and $A_{t+1} = A_t + \epsilon$. Everything else is more or less the same.

9.3 Incorporating the motion prior into the existing model

In order to run the particle filter, we just use the probability model described above. Now, the hidden state is the position and momenum of the particle. The initial probability is a gaussian prior. The transition probability is also a gaussian. For the likelihood function, we can probably just ignore the difference between X and $\hat{X}$, and use the spiking likelihood function after adding in the projection matrix to go from the full hidden state to just the position. For the proposal distributions, we can just use a matching gaussian proposal distribution (i.e. ignoring the current observation, as we did before).

9.4 Fitting the model to the data

Using the probabilistic model specified above, we can use the Kalman filtering algorithm to estimate $p(X_t|R_{0:t})$ or the Kalman smoothing algorithm to estimate $p(X_t|R_{0:T})$ where T is the total observation time. Since we have 3 parameters, we can just do a grid search to estimate log-likelihood for all of the parameters. (Perhaps some experimental knowledge can help us choose these parameters). A reasonable start for estimating σ_p is to just use the mean or RMS speed. I.e. we expect the eye to be moving at some speed, but we don't know the direction. The observation variance should be set by the experimental validation of the method of the eye tracker. The transition noise is there because our model doesn't perfectly model the actual situation.

9.5 Validating the Model

Supposing that we have done this model, how do we tell if it works? Probably the simplest way that we can do this is to use the actual eye motion paths and compare the simple diffusion model and the eye movement with momentum / acceleration models and see what does best in image reconstruction. In order to do this, we need to draw a correspondence between the parameters in the simple model and the new paramters. A simple idea to do this is to just fit the diffusion constant to the data in the same way as we did above. (If we remove the momentum term, we get the simple diffusion model).

9.6 Fitting Different Motion Models to Actual Trajectories

Given these models, we can find the log likelihood of the data under the model. Suppose that we are looking at one particular trajectory. We want to look at:

$$p(O_0, O_1, \ldots, O_T) = \prod_{t=1}^{T} p(O_t|O_{t-1}, \ldots O_0) \tag{107}$$

In order to calculate this using the quantities that we calculated in the Kalman filter, we have

$$p(O_t|O_{t-1}, \ldots, O_0) = \int dH_t \, p(O_t|H_t) p(H_t|O_{t-1}, \ldots, O_0) \tag{108}$$

So we can learn the parameters of the diffusion model and the velocity model on training data and compare the log-likelihood on some test data.

10 Drift in the presence of an inhomogeneous retina

Up to this point, we have modeled the retina as being completely homogeneous. Here, we are going to explore the hypothesis that drift is beneficial when the input array is not homogeneous. For instance, all cells have the same input-output relationship. Further, all receptive fields create a homogeneous array. In our framework, we can loosen these restrictions. We propose a few ways to diversify the cell array that can be incorporated into our model.

1. Jitter the positions of the different receptive fields by a small amount of noise. Eg. if the spacing between the receptive fields is a, add gaussian noise of $0.1a$ to each receptive field.

2. Diversify the cell types so that there are some ON RGCs and some OFF RGCs.

11 Common Questions and Subtle Concepts

When working on this research and discussing these ideas, it is evident that there are a number of challenging conceptual issues related to this work. In the following sections, I'll enumerate some of them.

11.1 Perceptually Relevant Cues versus Pixel Reconstruction

One important point that we have to keep in mind when modeling biology is that we don't literally reconstruct a pixel perfect image in our vision. For instance, when looking at a picture of white noise on a TV screen, it is impossible to remember the pattern on the screen. Thus it is important to keep in mind that our brain is seeking to estimate cues from the scene that help determine actions.

11.2 Parsimonious versus Biophysically Realistic Modeling when making predictions

11.3 Optimality of a coding strategy versus experimental test of the use of such a strategy

From previous work, it is evident that the human visual system performs better in the presence of fixational eye movements. The argument is that discrimination on a demanding 2 AFC task is diminished if fixational eye movements are compensated for. While this is an important step in the right direction, it doesn't actually explain why such eye movements are inherently useful. Eg. it could be the case that the next level of the visual processing stream gets unexpected input from the previous layer when fixational eye movements are compensated for and doesn't perform well for that reason. **?** discussion needs to be improved

Some work has suggested that the fixational eye movements spread power of the signal from purely the space domain into the space-time domain **?**. However, this is somewhat unsatisfying as we now need a decoding algorithm. Also it seems like one should be able to make this argument better in terms of optimal coding.

11.4 Biological Implementation versus Implementation suggested by the model

In the Burak paper, and in this paper, there seems to be an emergent way to implement the algorithm. Indeed, the Burak paper describes a what and a where population that interact in order to dynamically route the spikes incoming from the retina. Likewise our extension of their algorithm suggests such a neural implementation. However, just because we have one possible implementation, that doesn't mean that we have found the one that biology has chosen to use.

There is already some evidence to suggest that the actual biological situation may be more complicated. For instance, a fundamental assumption of this model is that there is one

global motion that is estimated and used to compensate for the motion of the eye. Recent experiments by Arathorn et al. have begun to address this assumption in **?**. The question is: if you project an image with a bounding box and a small box moving inside, how well can you see the relative motion of the box and the square? The results are somewhat complicated in the sense that you can see the relative motion, but your threshold for detecting motion is higher if the motion is consistent with the retinal slip. That is to say that there isn't some global motion estimation that subtracts out the motion.

Anyways these experiments suggest a type of computation where local motion is computed (i.e. a motion field) and this is used to compensate for the motion. It is unclear how exactly this type of computation would happen in a neural circuit. It is also unclear to what extent the higher level perceptual mechanisms impact the Arathorn et al. result.

12 Comparisons to Alternative Models

12.1 Burak Model

Indeed, as we are building an extension to the Burak model, it is worth considering what comparisons can be made to this model. As we will describe below, it is somewhat difficult to make a direct comparison.

Our model is more general in terms of space and the pixel intensities. Thus to make a comparison we would need to test our model for spikes that are generated using discrete space diffusion and binary pixels. Of course then the Burak model will perform better due to having stronger priors on these quantities.

Likewise, we cannot use the same error metrics (eg. the Burak model uses percentage of pixels correct whereas we would use something more like SNR.

Another issue is that the performance of these algorithms is highly dependent on the image size and the content of the image.

I am looking for good ideas of how to make a direct comparison (instead of a qualitative one).

12.2 Spike Averaging

If the stimulus is not moving at all, we can imagine that we just sum the spikes coming from a particular neuron and we can recover the image. This suggests a simple decoding mechanism where we just take an average of the spikes. Note that this method also relies on the fact that the neurons are ordered using a uniform grid.

Anyways, suppose that we have a rectangular grid of neurons indexed by j, whose responses as a function of time are $R_{t,j}$. The simplest thing that we can do is:

$$\hat{R}_j = \sum_t R_{t,j} \tag{109}$$

Likewise, instead of summing, we can convolve with a kernel that is more local in time. Eg.

$$\hat{R}_j(t) = \sum_{t'} K(t-t') R_{t',j} \tag{110}$$

An exponential moving average corresponds to $K(\Delta t) = \exp(-\Delta t/\tau)$ for $\Delta t > 0$, and zero otherwise. Note that this is a causal filter. Likewise, we could do a non-causal filter such as $K(\Delta t) = \exp(-(\Delta t)^2/\tau^2)$. Anyways, both of these methods are a reasonable point of comparison.

In the following argument, we will show that these arguments are doomed to fail. The basic problem is that in the time that we need to integrate over to determine the identity of the pixel, the image will have moved to the point of blurring our image. Notice that these methods are linear functions of our image, so we are free to look at images in any basis that we want. A simple basis is to look at images with a fixed spatial frequency in one direction.

Define the following problem: suppose that we have our simulations where the minimum firing rate is λ_0, the maximum firing rate is λ_1, the diffusion constant is D, and the image has a constant wavelength, eg. $I(x, y) = \sin \frac{2\pi}{\lambda} x$.

How much time, Δt, do we need to integrate over to distinguish a firing rate of λ_0 versus λ_1. The number of spikes in a time interval follows a poisson distribution with $\mu = \sigma^2 = \lambda_i \Delta t$. If we make a rough approximation that these poisson distributions are Gaussian, then we want to have these to distributions not overlap, which can be summarized by:

$$\lambda_0 \Delta t + k\sqrt{\lambda_0 \Delta t} < \lambda_1 \Delta t - k\sqrt{\lambda_1 \Delta t} \iff \frac{k^2}{(\sqrt{\lambda_1} - \sqrt{\lambda_0})^2} < \Delta t \tag{111}$$

where we want the means to be separated by k standard deviations. Thus we have an order of magnitude of how long we need to integrate to distinguish the minimum and maximum firing rate.

Since we have diffusion, we know that the image has moved approximately

$$\Delta x = \sqrt{D \Delta t} \tag{112}$$

Supposing that our image has a wavelength of λ, then we want the movement of the image to be small compared to λ. Eg. we want

$$\lambda > 4\Delta x = \sqrt{D \Delta t} \tag{113}$$

(eg. our shift is a fourth of the wavelength). Combining these equations, in order for our averaging decoder to work, we roughly want the following relation:

$$\lambda > \frac{4\sqrt{D} k}{\sqrt{\lambda_1} - \sqrt{\lambda_0}} \tag{114}$$

Using some typical numbers, we have $D = 100 arcmin^2/s$, $\lambda_1 = 100 Hz$, $\lambda_0 = 10 Hz$, $k = 2$ (standard deviations), we get $\lambda > 12. arcmin$. This is much poorer performance than the performance of the graphical model based approach. Furthermore, we note that this does not improve based on the size of the image, or over time. It will be interesting to see how well these scaling arguments apply to predicting the performance of these techniques.

12.3 Correlations

G. Kenyon has a paper on using the first eigenvalue of the correlation matrix of the spikes... Seems like it will have a similar problem with the averaging models... It is less clear how correlations will impact the reconstruction **?**. Need to understand basis for correlation matrix decoding better There is a lot going on in this paper that is unclear to me. Eg. they only look at fixational eye movements in their second method of simulating spike trains. It is unclear which figures use this data (only the last figure clearly indicates that FEM are included).

12.4 Comparing Diffusion Coefficients Across Different Models

One important but subtle point when doing comparisons across models is that the diffusion constant can be defined using different constants in different papers. In order to facilitate comparison, we give a simple way to match these coefficients across models. Regardless of the model, it should be the case that $\frac{E[(X(t+\Delta t)-X(t))^2]}{\Delta t}$ is the same constant across different models. Lets work out this ratio for a few models:

Burak Model: Diffusion happens on a rectangular lattice with lattice spacing $a = 1$. Each of the four possible steps happens with a probability $D\Delta t$. Thus the ratio is $\frac{4D\Delta t a^2}{\Delta t} = 4Da^2$. Note that the Burak model is a little handwavy in terms of units. In the model, $a = 1$ or $\frac{1}{2}$, and $D = 100$.

Kuang... Rucci Model **?**: Here, the model diffusion using the diffusion equation $u_t = \frac{1}{2}D\nabla^2 u$. In this equation, the variance as a function of time is $2D\Delta t$, so we get $2D$.

This paper: Here, we are using continuous space, discrete time diffusion where we assume that

Thus to compare these two models, we have

13 Comparisons to Experiment

A crucial part of this modeling effort is to make comparisons to experiment. We will outline a number of directions to go:

1. Given physiological parameters of the firing rate, we want to get a sense of what diffusion constants are suitable for recovering the image. Note that this is highly dependent on the nature of the image (eg. size, frequency content). Also note this only gives us a bound on D_C for encoder-decoders that do not use spike timing.

2. Contrast, frequency discrimination. Eg. given a forced choice task, what is the minimum contrast at which we can determine the orientation of a Gabor of a particular frequency (with a certain amount of noise). Many such results have been produced and we could reproduce them.

3. Gabor detection with eye movement compensation. Eg. **?**, they look at a person's ability to discriminate a gabor when the eye movements parallel or perpendicular to the axis of the gabor. It turns out that motion along the changing direction of the gabor leads to better discrimination. We can see if our model reproduces the idea that motion in a particular direction influences the quality of the reconstruction.

4. Note my model is consistent with an experiment done by K. Ratnam and A. Roorda here at Berkeley where people's ability to see small letters isn't diminished when one uses artifical eye movements with similar statistics to actual eye movements.

5. Likewise, it would be exciting to think of some testable predictions of the burak model (like more visual context helps us see a letter better)... Eg. an E surrounded by a box is easier to see than just an E... the context helps you detect motion.

6. K. Ratnam and A. Roorda noted that in their talks, people ask a lot of different questions about tricks that the brain could use to solve their 4 AFC rotating E experiment. It may be helpful to have this model to investigate some of the tricks that people are proposing for how the brain could solve the perception problem.

7. Building upon the 4-AFC rotating E experiment, one prediction from the Burak and related models is that if there is more visual context, then the form-motion separating algorithm will be better able to lock on to the target, and thus this should improve performance on the task. That being said, it is worth exploring the strength of this effect and the number of trials necessary in order to get a statistically significant difference.

14 Model Choice Discussion

To begin with, it is clear that the retina is incredibly complex and not only do we not have a full understanding of the retina, but trying to implement all the biophysical details that we do know is a challenge in of itself. A fundamental assumption of this work is that we can make useful predictions without going into full biophysical detail. I.e. some properties of the problem are robust to some of the details of the retina. As we incorporate more properties of the retina into our model, we may be able to tackle certain prediction problems that are dependent on those details. In the following section, we enumerate some our thoughts on these issues.

14.1 Image to Spike Encoding Model

The choice of our image to spike encoding model is primarily driven by the fact that we can use an approximate version of Bayesian estimation to decode the spiking code that the model produces. We rely in previous research to explain why this model will allow us to capture important parts of the problem.

Most generally, we adopt the GLM modeling framework of the retina. To review previous work on this topic, let us describe the model from **?**. In this model, we assume that the spikes are generated by an inhomogeneous Poisson process where the rate parameter is given by the following:

$$\lambda(t) = \exp(k \cdot x + h \cdot y + \sum_i l_i \cdot y_i + \mu) \tag{115}$$

where x is the stimulus, y cell's own spike train history, and y_i is the spike train history of another cell, i, and μ is the log of the baseline firing rate. The stimulus x is a function of space and time, and thus we also consider the stimulus filter, k, to be a function of space and time. y and y_i are purely functions of time, and thus the corresponding filters h and l_i are also functions of time. k ends up giving a biphasic response profile. The filters h gives an inhibitory effect for short times and no effect for long times after a spike. The sign of the

filters l_i depend on a variety of factors including whether the cell is an ON or OFF RGC. An important paper that studied the predictive power of these modeling assumptions is **?**.

Given this rough frame work, we can think about what aspects of the model we want to incorporate and how those choices impact the predictive power of our model.

In our modeling, we are neglecting the coupling terms between the different neurons. (eg. the l_i terms). The main reason for this is that while we know that this is an important part of modeling the actual retina, it seems like modeling these connections adds additional complexity that isn't particularly important for making our point. Relatedly, it is hard to think of a principled approach to measure the coupling between all RGC cells in a patch of the fovea.

Next, we also neglect the post-spike filter. Here, we are just looking at modeling cells in terms of having high or low firing rates and it doesn't particularly seem that important to include these effects. (I.e. post spike inhibition, then small excitation, followed by no effect).

Finally, we consider the problem of the stimulus filter. This is a place where we also make substantial simplifications that may be somewhat important. First, we make a rough approximation that retinal ganglion cells are in one to one correspondence with cones in the fovea. While this is a reasonable first step, we know that this is not true in reality. Further, we assume that neurons respond to the instantaneous stimulus presented to the neuron instead of a biphasic integration curve. This last issue has been investigated in some depth in previous work. In Fig. 4 of **?**, they show that this type of integration curve makes the problem more difficult to solve the decoding problem. Eg. they show that if we know the motion of the retina, then the time to reconstruct the image increases substantially above the limit set by the poisson spiking for larger diffusion coefficients. We are actively looking into a better understanding of this effect and the relevance to our investigation. Another problem with this line of research is that the biphasic response curve was derived by reverse correlation with white noise stimuli and recent work has shown that this type of model fails to predict RGC activity in the case of natural stimuli.

Taking a step back, it is important to note that the GLM modeling described above is phenomenological. For instance, there is a layer of bipolar and horizontal cells between the retina and the RGC layer that does some sort of processing on the signals already. Relatedly, there are many different classes of RGC cells that we don't fully understand. The GLM framework models these connections as effective inhibition and excitation between RGC cells. One could use a more complex simulation of the retina like the J.H. van Hateren model that takes into account the connections between cells. Integrate and fire type models are more realistic biophysically but I am unaware of ways to decode spikes generated from a IF model in a princpled way.

With all of this being said, our hope is that we can say something useful about a system designed to use statistical dependencies between cells generated by a moving stimulus in order to decode the image (as opposed to statistical dependencies generated by the circuitry of the retina).

14.2 Eye movement modeling

Another source of potential model generality-parsimony is in the modeling of the fixational eye movements. There are still many open questions about fixational eye movements eg. **?**. In this work, we are trying to create a reasonable model of tremor and drift in the eye. These eye movements take a 20-20 sized letter and move it the size of the letter in a fraction of a second. The Burak model assumes simple diffusion on a lattice. One important advance in our work is finding a way to move to continuous space diffusion. In order to facilicate simulation, we use discrete time and each 'jump' is generated by a normal distribution with variance that is proportional to the time step. Eg. $T(x'|x) \sim N(\mu = 0, \sigma^2 = Ddt)$ where D is a diffusion constant.

One reasonable and important discussion worth considering is that drift seems to move the eye relatively smoothly compared to diffusion. It would be possible to include this effect by adding a momentum term for the eye movement. We hope to look at some experimental trajectories and it seems that this is an important effect to keep in mind.

There is a fair amount of other work on creating phenomenological models of the retina (eg. **?**), but they don't lend themselves well to probabilistic decoding.

14.3 Decoding Method

Given the set-up of the probabilistic graphical model, it is now up to us to decide on an inference techinque. As above, we generate data by the following process: Choose an image S. Generate a path $X = X_{0:T}$ according to the graphical model (jumps are normally distributed). Generate spikes $R = R_{0:T}$ according to a Poisson model where the image-receptive fields overlaps are calculated as above. The end result of the data-generation is that we have some spikes, R. Our goal (and to some approximation the goal of the visual cortex) is to use R to estimate S and X.

In the following, we will be roughly answering the question, given the noise associated with the Poisson spikes and movement of the image, how much information is contained about S and X in the spike train? Even if the brain doesn't use our particular algorithm, we can use Bayesian optimal estimation (eg. optimal if our graphical model is correct) to see how much information is left about the image. Note that our our model neglects all potential information that could be stored in spike timing.

A full Bayesian treatment is impossible for reasonably sized images. Eg. calculating $P(S, X|R)$ for all S and X. This is impossible because we need a number for each image and position, and there are, even in the binary image case, exponentially many possible images. Thus to pursue this line of thinking further, we need to do some sort of approximation.

The Burak model uses a sort of variational Bayes approximation where we approximate $p(X_t, S|R_{0:t})$ as $p(X_t|R_{0:t}) \prod_i p(S_i|R_{0:t})$. I.e. we assume that the distribution factors. The variational princple gives us a way to calculate recursion equations for these variables. This is called a variational (or mean field) approximation because we use the calculus of variations to find the optimal values for these factors by minimizing the KL divergence between the actual distribution and our factorized estimate. Eg. $p^{opt}(S_i|R_{0:t}) = E_{S_j \neq S_i, X} p(X_t, S|R_{0:t})$.

We can develop a recursion for these quantities by going back one step in the HMM. In general, this expectation is impossible to calculate because of an expectation over all images. However, given the factorized approximation and an *independent pixel approximation*, the sum factorizes (in a weird way) and these recursions can be computed. See the **?** for more details. (This expectation is also impossible to calculate reasonably efficiently if we want to consider continuous valued pixels and positions).

Anyways, a crucial problem that we want to address is that images aren't composed of independent pixels. It turns out that the fortuitous factorization that lets the variational bayes approximation work fails when you add in a coupling between pixels. Thus we need to look to other approximations to try and do the estimation. The simplest approach that one can do is to do MAP estimation. Eg.

$$\hat{X}, \hat{S} = \operatorname*{argmax}_{X,S} \log p(R, X, S) \tag{116}$$

Here, we replace looking over all possible images and paths with a point estimate of each of these. The long and short of it is that this doesn't seem to work well in practice. See later for more discussion. One important point there is that the algorithm doesn't take into account the sequential nature of the graphical model.

Anyways, the next more complicated approach is to do expectation maximization. Here, we seek to estimate

$$\hat{S} = \operatorname{argmax} p(S|R) \tag{117}$$

See the sections on expectation maximization to see the details of this approach.

15 Lessons Learned from Bayesian Inference

When you start to seriously think about the problem of extracting a moving image out of a noisy signal, you realize that it is actually quite difficult. Naive methods seem to have poor performance and a principled approach is much more satisfying in terms of trying to understand the problem. Intriguingly, the Bayesian probabilistic approach gives us a natural way to go about this problem. If we can specify the encoding model that takes us from an image to spikes, then we can just follow the math in order to get the optimal decoder. This method gives us a reasonable starting point for understanding what types of computations and considerations should be present in trying to solve the problem at hand. Here we outline some of the results that line of reasoning:

15.1 Dynamic Routing of Spikes to form Image Estimate

Perhaps one of the most straightforward results of this work is that supposing that we have an estimate of the position of the image after some time has passed, we can use that estimate to dynamically route the incoming spikes from a particular neuron to the correct part of the internal representation of the incoming image. This idea isn't particularly new. For instance, it is common to think about the efferent copies of motor commands to the eye muscles correcting for the eye movements during larger eye movements such as saccades. The difference in this situation is that we estimate the motion directly from the image instead of having an efferent copy of the motion command to facilitate dynamic routing. Research suggests that the efferent copy correction doesn't make sense in our context because the brain doesn't have access to movements at the resolution necessary to correct for drift-sized eye movements Refs from burak introduction... This point is somewhat contested.

15.2 Propagating the position estimates

While the estimate of the image given the position is relatively straightforward, the method to estimate the trajectory given the image might be less obvious. In some senses, the idea is congruent with using a Kalman filter (or another continuous space hidden markov model) to estimate the motion of an object given noisy observations of position. The section on the discrete space EM in real time gives us the best view on this issue. The basic idea is that we have some current estimate of the location of the retina with an associated uncertainty. During the forward propagation step, we use our prior knowledge of how the image moves to propagate that estimate (crucially, with uncertainty). As we don't have a perfect model of the motion, this step increases our uncertainty. Next, we look at the likelihood of the incoming spikes given our estimate of the image for different positions, and use that to reduce our uncertainty in our position estimate. The probabilistic model is crucial here in giving us a concrete way to weight our estimates of the position of the eye after new observations come in. Methods like naive averaging and correlation analysis don't have a good answer to the question of estimating the position.

15.3 Propagation of uncertainty is crucial in the presence of ambiguous signals

One goal of our algorithmic work is to generate a real-time (or at least time windowed) approach generating the estimates of the image and motion. It is somewhat implausible that the visual cortex would memorize the incoming spikes for a substantial duration of time. So when we go about this route, we see that a parsimonious representation of the incoming data can be formed by both saving a point estimate of the image and the associated uncertainty. This way we are not storing exponentially many probabilities for each possible image. The reason that storing uncertainty in addition to the mean estimate is that when we get a new set of observations, we need a principled way of weighting our confidence in the previous estimate with the estimate that the new data provides. This is critical in environments where any particular snapshot gives you relatively little useful information. This view is commonly discussed in the context of quantifying one's uncertainty in Bayesian statistics.

15.4 Details of the Image Representation are important

An important issue that hasn't quite been clear in the discussion thus far is that the representation of the image is important. When we go from a continuous world to a discrete retina, we need to be careful about the choice of our representation. Needs more info here

16 Additional questions to investigate

1. If we diversify the neuron cell types, we should get better performance in certain regimes of noise. **?**... perhaps there are better citations for this issue. The most obvious example is of ON and OFF retinal ganglion cells. Another important example is the wavelength-dependent inhomogeneity of the cone lattice.

2. Model robustness. If we start providing the model with input generated using a different model, how well does it perform?

3. Add another layer to the model to do AFC tasks... Eg. after estimating the stimulus, compare it to a set of templates in order to try and guess the image.

17 MAP Estimation Results

It seems to be the case that the MAP estimation doesn't work very well. Roughly speaking, my idea for why this is the case is that we only have good position estimates for very few time points, so the gradient to estimate the image is very noisy.

17.1 Expectation values of the MAP Method

In order to help guide our MAP analysis, it is worthwhile to look carefully at the expected value of the function that we are trying to maximize. In particular, suppose that we draw a sample from our model, X', R, S'. What does the objective function look like?

$$\sum_t -\log p(X_t|X_{t-1}) \approx N_T \cdot E[-\log p(X_1|X_0)] \tag{118}$$

where N_T is the number of timesteps. Now $p(X_1)$ is gaussian distributed with mean X_0, so we get

$$E[-\log p(X_1|X_0)] = E_{p(X)} \frac{1}{2D_C\Delta t}(X)^2 = \frac{1}{2} \tag{119}$$

where $p(X) \sim N(0, \sigma^2 = D_C\Delta t)$. Now lets do a similar analysis for the spiking term. However, we will additionally look at the situation where we get the image incorrect. In particular, suppose that the actual pixel presented to a neuron has an intensity s^* that results in a firing rate of λ^*. Suppose that we erronously estimate this intensity as s with a firing rate λ. What is the value of our spiking log-likelihood going to be?

$$\sum_{j,t} -\log p(R_{j,t}|X_t, S) \approx N_T \cdot N_N \cdot E[-\log p(R_{j,t}|X_t, S)] \tag{120}$$

where the expectation is taken over the actual image S^*, and N_N is the number of neurons. This expectation is then equal to (depending on whether we have a spike or not)

$$-\log(\lambda\Delta t) * (\lambda^*\Delta t) + -\log(1-\lambda\Delta t)(1-\lambda^*\Delta t) \tag{121}$$

Now let us calculate the difference between this function when we get it wrong $\lambda = \lambda$ and when we get it right $\lambda \to \lambda^*$. The point here is to see how much the objective function changes when we guess versus we get the right pixel intensity. We get

$$-\log\frac{\lambda}{\lambda^*}\lambda^*\Delta t - \log\frac{1-\lambda\Delta t}{1-\lambda^*\Delta t}(1-\lambda^*\Delta t) \approx \Delta t \cdot (\lambda - \lambda^* + \lambda^* \cdot \log\frac{\lambda^*}{\lambda}) \tag{122}$$

Note that expression is non-negative with a minimum when $\lambda = \lambda^*$. FIXME: I think we are calculating the mutual information or some info theory related quantity.

So supposing that $T = N_T\Delta t$ is the total time interval, then the costs for these two terms are:

$$\frac{T}{2\Delta t} + N_N \cdot T \cdot (\lambda - \lambda^* + \lambda^* \cdot \log\frac{\lambda^*}{\lambda}) \tag{123}$$

Notice that as Δt goes to zero, this expectation goes to infinity. Thus it would seem that if we chose the path to be all zero and fail to reconstruct the image, then we would get a lower energy. However, the resolution to this paradox is that the optimal solution is smoother than the average solution afforded by the cost function.

17.2 Rate Coding as dt goes to zero

As in the L. Paninski papers, we can take the limit as dt goes to zero and recover a different form of the log-likelihood. Suppose that we have some spike train $r_{j,t}$ and that there is some instantaneous firing rate function $\lambda_{j,t}$ where j denotes neuron number and t corresponds to time. Then the log-likelihood assuming time windows of dt that are sufficiently small so that there is only one spike in each is

$$\log p(R) = \sum_{j,t} R_{j,t} \log \lambda_{j,t} dt + (1 - R_{j,t}) \log(1 - \lambda_{j,t} dt) \tag{124}$$

Now we can take the limit as dt goes to zero. $\log(1 - x) = -x + O(x^2)$, so when we expand out the summand (and dropping the index j for convenience) we get:

$$R_t \log \lambda_t + R_t \log dt + (1 - R_t)(-\lambda_t dt + O(dt^2)) \tag{125}$$

Next, we sum this quantity over t. Note that we have a finite number of spikes in any time window of size T. So $\sum_t R_t \lambda_t dt \to 0$. Further, $\sum_t R_t \log dt < T \frac{\log dt}{dt} \to 0$. Thus we are left with:

$$p(R) = \sum_j \left(\sum_t R_{j,t} \log \lambda_{j,t} - \int_0^T dt \, \lambda_{j,t} \right) \tag{126}$$

17.3 Rate coding with one neuron and one pixel

In this section, I present a calculation that tells us what happens if we try and estimate one pixel using a rate code. Suppose that we have a pixel of intensity s_0 and a known intensity to firing rate function $\lambda(S)$. If we run a simulation for a total time, T, how well can we estimate S_0? Suppose that we discretize time into time intervals dt and we observe a spiking pattern R. Then we can use the equation derived in the previous section to get

$$\log p(R|S) = N(T) \log \lambda(S) - T \cdot \lambda(S) \tag{127}$$

where $N(T)$ is the number of spikes in the time interval. Denote λ^* as $N(T)/T$. This should be approximately constant, and then we note that $p(R|S)$ is of the form $\exp(T \cdot f(S))$. If we look at the limit as T goes to infinity, we see that this is asymptotic to an expansion of $f(S)$ about its maximum.

$$\frac{\log p(R|S)}{T} = \lambda^* \log \lambda(S) - \lambda(S) \tag{128}$$

First, the maximum of this function occurs when $\lambda(S) = \lambda^* \rightarrow S = \lambda^{-1}\lambda^* \equiv S^*$. Thus the function is a constant plus the second term in the Taylor series expansion:

$$\log p(R|S) = C - T\lambda(S^*) * \left(\frac{\lambda'(S^*)}{\lambda(S^*)}\right)^2 (S - S^*)^2 \tag{129}$$

From here, we can see that for a time interval T, we have, asymptotically,

$$p(R|S) = N(\mu, \sigma^2) \qquad \mu = \frac{N(T)}{T} \qquad \frac{1}{\sigma^2} = T\lambda(S^*) * \left(\frac{\lambda'(S^*)}{\lambda(S^*)}\right)^2 \tag{130}$$

where we had defined $N(T)$ as the number of spikes in the time interval T, $\lambda^* = \frac{N(T)}{T}$, and $s^* = \lambda^{-1}\lambda^*$.

What we can see from this result is that we can most accurately estimate S when $\frac{d}{dS}\log\lambda(S)$ is large at $S = S^*$, or $\lambda'(S)$ is large compared to $\lambda(S)$ at $S = S^*$.

Further, we note that $N(T)$ is a poisson process, and the number of events that occur in an interval has a poisson distribution with mean λT, i.e. $P(N(T) = k) = e^{-\lambda T}(\lambda T)^k/k!$. We note that

$$E(N(T)) = \lambda(S_0)T \qquad \mathrm{Var}(N(T)) = \lambda(S_0) \tag{131}$$

If λ changes in time, we replace $\lambda T \rightarrow \int_0^T dt\lambda_t$.

It is also worth noting that if we are trying to estimate J different pixels, then we can use this estimate to look at the joint

17.4 Normalization

One important aspect that may be contributing to problems in the simulation is the problem of normalization of the input image. As the calculations above show, $\lambda(S)$ has to be chosen carefully. As in the work of Paninski et al., if we choose a nice form of λ, then the log-likelihood that we are optimizing is convex as a function of s. In particular to do this, it is sufficient to have $\log\lambda(S)$ concave and $\lambda(S)$ convex. One problematic consequence of the concavity of $\lambda(S)$ is that if we have an increasing function, then it becomes unbounded. We can handle this issue by adding spike history effects to keep the firing rate bounded (see below). (Paninski 2004).

18 Corrections

List of Corrections

almost copied from Burak et al intro, needs to be rewritten or cited properly . . . 4
Add refs for image priors . . . 4
Think about how to explain mismatch between image and latent factors . . . 27
discussion needs to be improved . . . 33
Need to understand basis for correlation matrix decoding better . . . 36
Refs from burak introduction . . . 42
Needs more info here . . . 43